Week 8: Metrics in SE for SPM

Rachelle Bosua

rachelle.bosua@unimelb.edu.au

www.lms.unimelb.edu.au

Outline

- Intro and recap
- Measuring complexity in software
  - Cyclomatic complexity - *done*
  - CK metrics suite ✓
  - Function point analysis ✓
    - All the steps of function point

Metrics to Measure Complexity

Three methods...

1) Cyclomatic complexity
2) The CK metrics suite
3) Function point analysis

The CK metric suite

To measure **OO design complexity**
(Chidamber & Kemerer 1994)

6 Measures:

1) Weighted methods per class (WMC)
2) Depth of the inheritance tree (DIT)
3) Number of children (NOC)
4) Response for a class (RFC)
5) Coupling between Object Classes (CBO)
6) Lack of cohesion in methods (LCOM)

CK-1) WMC

Measures complexity of each method in a class

How?

- *Estimate* if source code/internal design not available
- If a class has as number of methods: n and complexity measure: $c_1, c_2, c_3, \ldots c_n$
- *WMC for class is:*

straight summation $\sum_{i=1}^{n} c_i$

Minimize WMC value per class

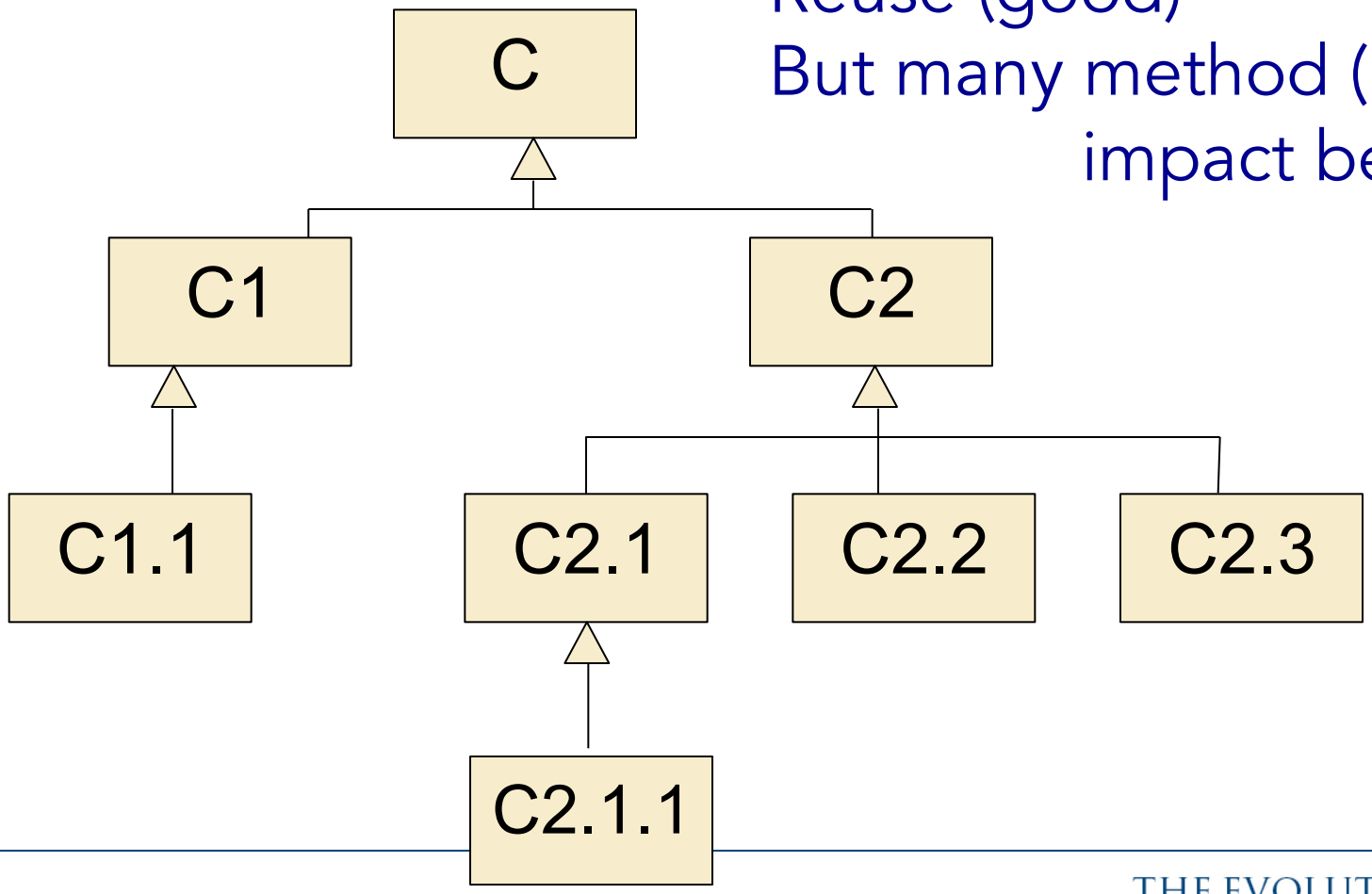
CK-2) DIT

The maximum length path from the root class to a leaf no (class without any subclasses) in an inheritance tree

- The deeper a class in the hierarchy
  - a likelihood of being more complex
- Deeper trees constitute more design complexity, as more methods & classes involved
- The deeper a particular class, the greater the potential reuse of inherited methods

CK-2) DIT

DIT = ?
High DIT positive and negative
Reuse (good)
But many method (high WMC)
impact behaviour



CK-3) NOC

Count of the number of ***direct*** **children** of a class

CLASS	NOC
C	?
C1	?
C2	?
C2.1	?

Other classes has an NOC of zero

High NOC can be both positive and negative.
High NOC = reuse
But parent class abstraction might be diluted

CK-4) RFC

= for a method in a class the number of method calls made by that method **if invoked**

So the summation of the size of the response set for all methods in a class

- A class with *n methods* and response set $r_1, r_2, r_3, \ldots r_n$ for each method the
- *RFC of the class is:*

$$\sum_{i=1}^{n} \neq r_i$$

High RFC – higher complexity
Need implementation of each method to estimate the no of method calls

CK-5) CBO

= for each class the number of relationships the class has with other classes excluding inheritance, i.e. such as aggregation and association

- Class A is *coupled* with Class B if either of them 'act upon' each other
- A class with a high CBO is **less likely to be reusable** than one with a low CBO
- A high COB value make maintenance and testing of class **more difficult**

CK-5) LCOM

LCOM value = the no. of pairs of methods whose similarity is zero minus the no. of pairs of methods whose similarity is not zero

- E.g. methods m_1, m_2, m_3 access attr/variables:

$$m_1 = \{v_1, v_2\}$$
$$m_2 = \{v_2, v_3\}$$
$$m_3 = \{v_4\}$$

All possible Cartesian pairs of methods in the class and attributes the share are:

$$m_1 \cap m_2 = \{v_2\}$$
$$m_1 \cap m_3 = \{\}$$
$$m_2 \cap m_3 = \{\}.$$

Cohesion of the method is 2 – 1 = 1

CK-6) LCOM

- Higher LCOM value is 'bad'
- Good design should have a high cohesion
  - i.e. a class method should access the same attributes as each other, otherwise they are *unrelated*
  - If a method accesses only ONE variable and that variable is only accessed by that method (e.g. method m_3 & variable v_4) → perhaps the method and variable need to be refactored into another class

Metrics to Measure Complexity
(3) Function Point Analysis

Function Point Analysis

The *most* widely use metric to measure the complexity of requirements

Definition:

A unit of measurement that to express the amount of functionality in a SW system, as seen by the users

- A high number of function points indicates MORE functionality
- *Positive correlation*: function points & the complexity of the system

Why using Function Points?

1) Estimate **cost ($$) and effort** to design, code and test a software system
2) Predict the **number of errors**
3) Predict **number of components** in software

5-step calculation process

1) From user requirements, ***categorize each functional req's*** in 1 of 5 cat's
2) ***Weight complexity*** of each category
3) Calculate a count total
4) Calculate a weighting - *value adjustment factors*, to **weight** the NF req's
5) Calculate ***the total FPoints count***

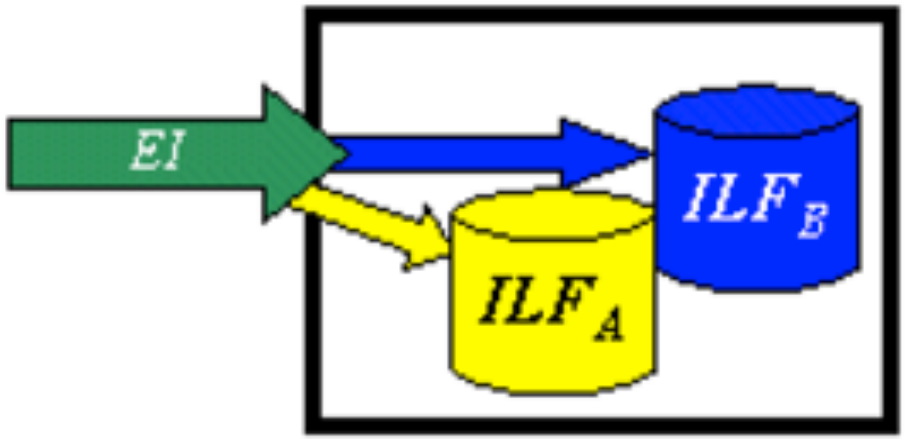
Revisiting each **Function Point Analysis** step

1) Categorizing functions..1

From user requirements, *categorize* each functional req's in **1 of 5 different categories** based on inputs, processing and outputs related to data

1) Categorizing functions..1

Internal Logical File (ILF) – data
a user identifiable group of logically related data that resides entirely within the application boundaries and is maintained through external inputs



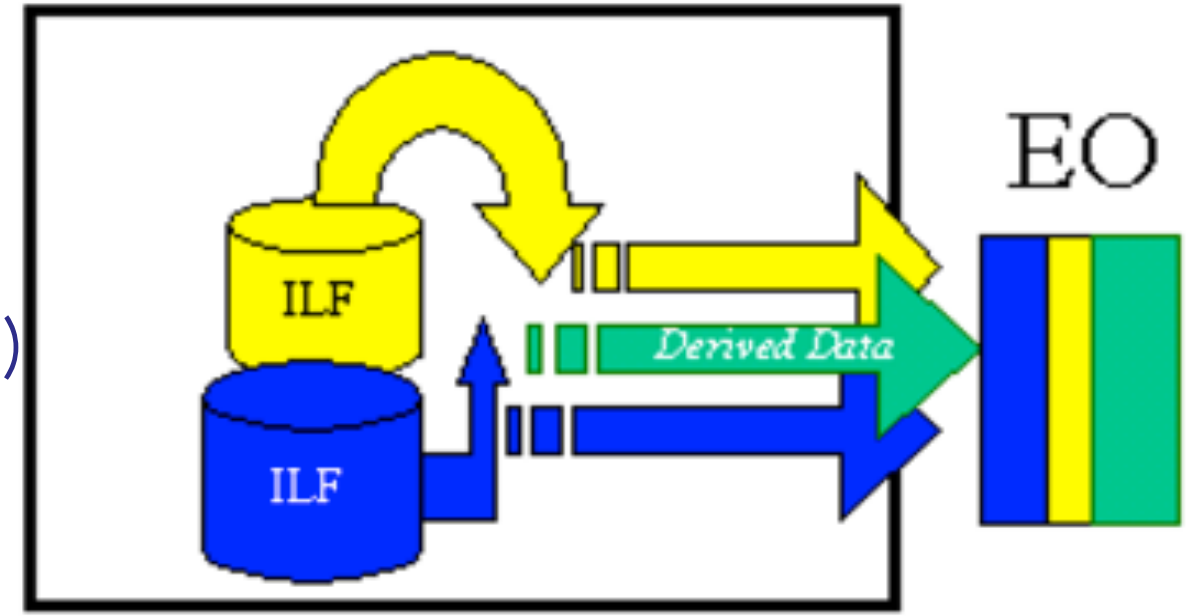
External Input (EI)
- transaction an elementary process in which data crosses the boundary from outside to inside. This data may come from a GUI or another application & may be used to maintain one or more internal logical files

1) Categorizing functions..1

External Outputs (EO) - transaction

an elementary process in which derived data passes across the boundary from inside to outside. An EO may update an ILF. The data creates reports or output files sent to other applications

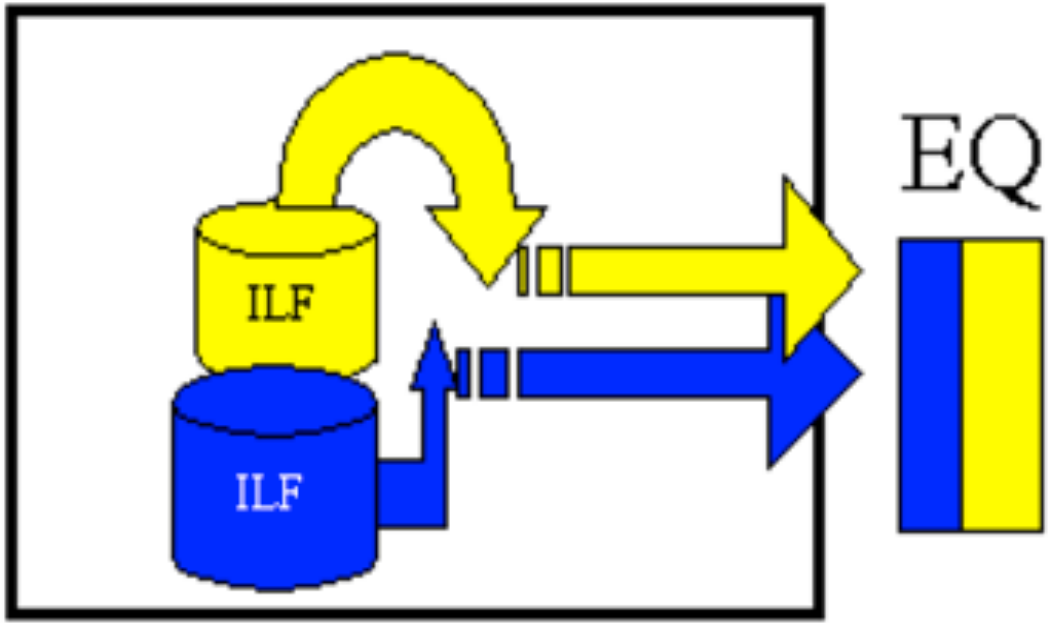
This is an EO with 2 FTR's There is derived Information (green) derived from the ILFs



1) Categorizing functions..1

External Inquiry/Query (EQ) – transaction
is an elementary process with both input & output components that result in data retrieval from one or more internal logical files and external interface files

The input process does not update any ILFs and the output does not contain derived data



1) Categorizing functions..1

External Interface File (EIF) - data

logical grouping /or user-identifiable data/control data that is used for reference purposes only. The data resides totally outside the application and is maintained by another application – so the system maintains data outside the system (e.g. data hosted by 3rd party)

So the external interface file is an internal logical file for another application

2)Assigning complexity val's

- Once classified *assign* ***complexity value*** associated with each category
- Rank **category complexity** as: *simple/low, average* or *complex/high*
- Rate all EI's, EO's, EQ's ILF's or EIF's as (L,M,H)
- EI's, EO's and EQ's ranking = # of files updated or references (FTR's) and # data element types (DETs)
- For ILF's and EIF's files ranking is based on record element types (RETs) and data element types (DETs)

2)Assigning complexity val's

- Use tables to do a ranking -

EI Table

FTR's	DATA ELEMENTS		
	1-4	5-15	> 15
0-1	Low	Low	Ave
2	Low	Ave	High
3 or more	Ave	High	High

Shared EO and EQ Table

FTR's	DATA ELEMENTS		
	1-5	6-19	> 19
0-1	Low	Low	Ave
2-3	Low	Ave	High
> 3	Ave	High	High

Eg. an EI that references/updates 2 File Types Referenc'd (FTRs) and has 7 data elements=Ave

2)Assigning complexity val's

- Use tables to do a ranking -

Values for Transactions

Rating	VALUES		
	EO	EQ	EI
Low	4	3	3
Average	5	4	4
High	7	6	6

RET's	DATA ELEMENTS		
	1-19	20 - 50	> 50
1	Low	Low	Ave
2-5	Low	Ave	High
> 5	Ave	High	High

2) Assigning values..2

Rating	Values	
	ILF	EIF
Low	7	5
Average	10	7
High	15	10

Note here if there are

Counts & Complexity table

Enter counts and complexity in a table, multiply count by rating to determine rated value

Type of Component	Complexity of Components			
	Low	Average	High	Total
External Inputs	___ x 3 = ___	___ x 4 = ___	___ x 6 = ___	
External Outputs	___ x 4 = ___	___ x 5 = ___	___ x 7 = ___	
External Inquiries	___ x 3 = ___	___ x 4 = ___	___ x 6 = ___	
Internal Logical Files	___ x 7 = ___	___ x 10 = ___	___ x 15 = ___	
External Interface Files	___ x 5 = ___	___ x 7 = ___	___ x 10 = ___	
			Total Number of Unadjusted Function Points	___
			Multiplied Value Adjustment Factor	___
			Total Adjusted Function Points	___

Use one of 14 general system characteristics that rate general application functionality

Benefits of Function Points

- Accurately size SW applications – sizing is important for productivity
- Good to manage scope creep
- Basis for estimating models
- Used with other metrics
- Communication
- Useful across teams and roles
- Easily understood by a non-technical user

Ref:
http://www.softwaremetrics.com/fpafund.htm

Cost and effort estimation

- Accurate estimates of effort and cost
- Guessing to machine learning techniques
- Parametric models – low cost + accurate!

WHAT IS PARAMETRIC?

References

- Chapter 7 Notes (of SWEN90016)
- Function Point Analysis: http://www.softwaremetrics.com/fpafund.htm
- https://www.tutorialspoint.com/estimation_techniques/estimation_techniques_function_points.htm
- https://www.tutorialspoint.com/estimation_techniques/estimation_techniques_function_points.htm
- Pressman RS (2002) *Software Engineering a Practitioner's Approach* – 5th Edition, McGraw-Hill